Yes, yet another PHP 5 installation guide. Why write one when there are already dozens of them out there? Well there are two main reasons: firstly I will be writing a lot of guides that require PHP to be installed and working and I will need something to point people to who have yet to install PHP. Secondly, I have read a lot of the guides out there and found that a good portion of them over complicate the process. It really should not be that hard or perplexing. In this guide we'll break down the process of installing PHP 5 on IIS into 5 simple steps that are easy to follow that works for **IIS 5**, **IIS 5.1** and **IIS 6**.

NOTE: This guide is intended for use with PHP 5.2.x and below. A guide for PHP 5.3 and above will be published in the coming weeks.

So what are these 5 simple steps? In a nutshell, this is all you need to do to install PHP 5 on IIS;

1. Download and Extract PHP 5 (we'll install PHP 5.2.1 on IIS 5.1 in this guide)
2. Set your environment variables
3. Set the PHP application mapping in IIS
4. Configure php.ini
5. Restart & Verify

Just before we continue I thought it pertinent to mention that it if you have already installed, or attempted to install PHP on your system then you must remove all traces of that install before continuing. If ANY PHP files exist anywhere else on your system that are accessible via the PATH environment variable, then it is very possible (even probable if the previous version of PHP was different) that you will have issues with hosting PHP content.

If you are wanting to install PHP 4, then the process is slightly more complex than installing PHP 5 and involves steps not covered in this guide. Not to worry as it is even easier for you. All you have to do is download the IIS Aid PHP Installer from this site and run the fully automated GUI installer. This installer will completely setup your PHP 4 environment, including all environment variables, IIS scriptmaps and IIS 6 web extension. The particular advantage of this installer is that it configures PHP 4 to use FastCGI mode as well as an Opcode cache for maximum performance. Expect to see some articles of just how much faster here in the future.

1. Download and Extract PHP 5

Nothing taxing about this, simply go to the PHP download page and download the latest **PHP zip package** from under the "Windows Binaries" section, along with the latest **PECL Win32 binaries** (do **not** get the ones marked "non-thread-safe" unless you plan to use CGI/FastCGI as otherwise you run the risk of a race condition). Save the PHP zip package to the directory where you will want to install PHP (e.g. D:\PHP), and then extract. Once extracted you'll see inside your PHP installation directory a sub directory called "ext". Save the **PECL Win32 binaries** download into it (e.g D:\PHP\ext), and then extract. Your PHP install directory should now contain all the content of your **PHP zip package** file, and your ext sub directory should contain the contents of your **PECL Win32 binaries** file. From here move back to the root of you PHP installation directory (e.g. D:\PHP) and create three directories, one

called **sessions**, one called **upload** and one called **log**. In the **log** directory create a new file called **php.log**. First step licked.

2. Set Your Environment Variables

For this step we'll need to use the Windows GUI, so rather than just write out all the steps I'll use images to do most of the talking.

- Right click on the **My Computer** icon and select the **properties** menu item. You should be presented with a window as shown below in figure 1. Select the **Advanced** tab and then click the **Environment Variables** button at the bottom.

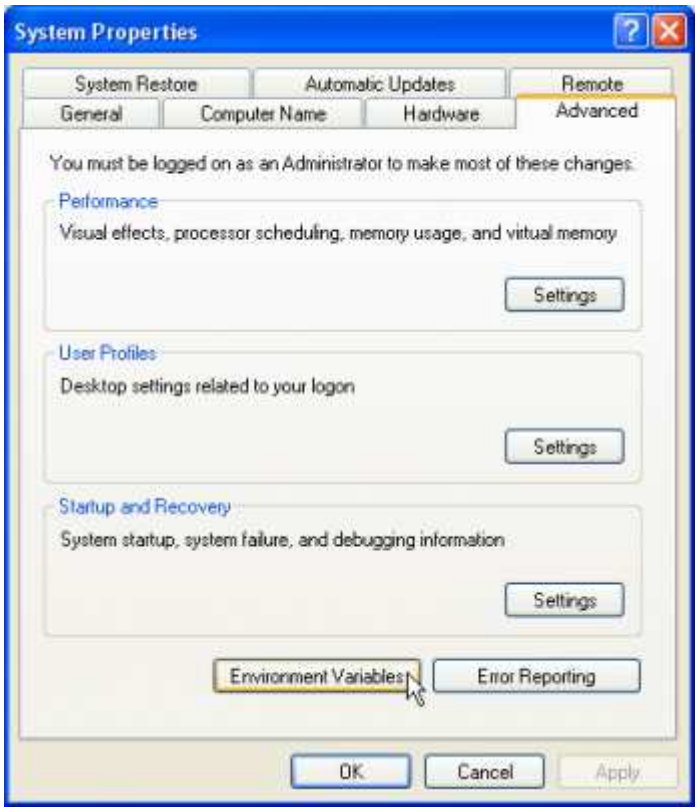

Figure 1

- Scroll through the **System variables** list at the bottom until you come to the **path** variable. Select it and then click the **edit** button as shown below in figure 2.

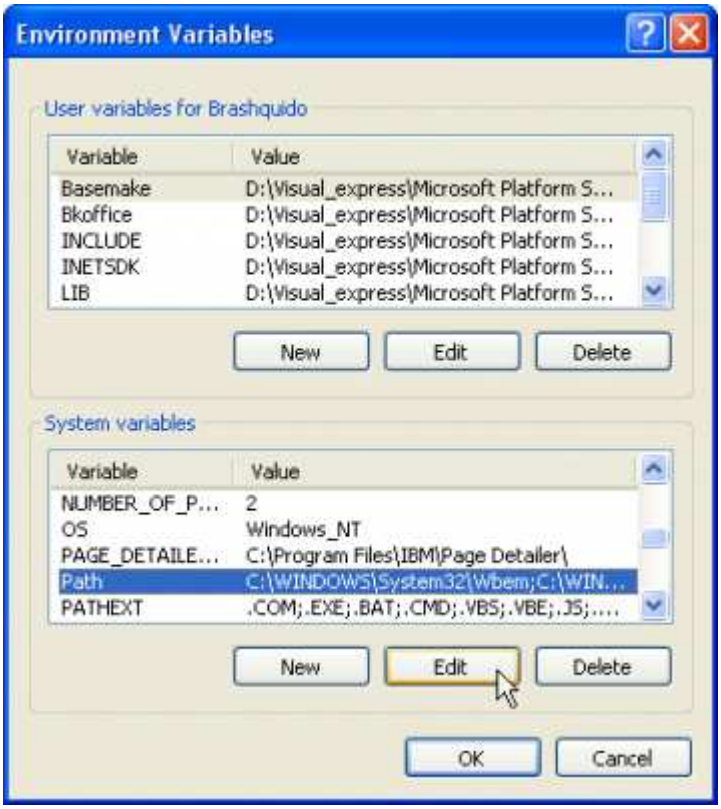

figure 2

- Move to the end of the text in the **Variable value** field and type in the path to the PHP directory you created earlier in step 1 (e.g. D:\PHP). Make sure that there is a semi colon (;) separating the new value you're entering and the previous one, and also place another semi colon (;) after as shown below in figure 3. Once you have added the path click on the **ok** button to save it and return to the previous window.

figure 3

- This time we need to create a new environment variable. From the **Environment Variables** window click on the **New** button towards the bottom as shown below in figure 4.

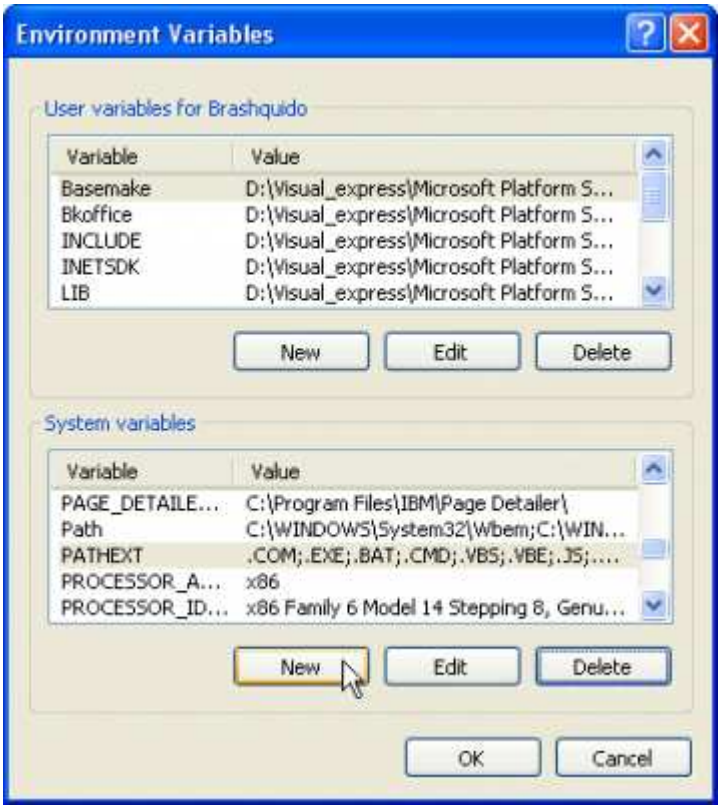

figure 4

- You should now see the **New System Variable** window as shown below in figure 5. In the **Variable name** field type in **PHPRC**, and in the **Variable value** field type in the path to your PHP install directory (e.g D:\PHP) as shown below. Once done click the **ok** button to create and save the variable. This **PHPRC** variable tells Windows where to look for your php.ini file, and if it doesn't find it there or if this variable is not set then IIS will look though your Windows system path.

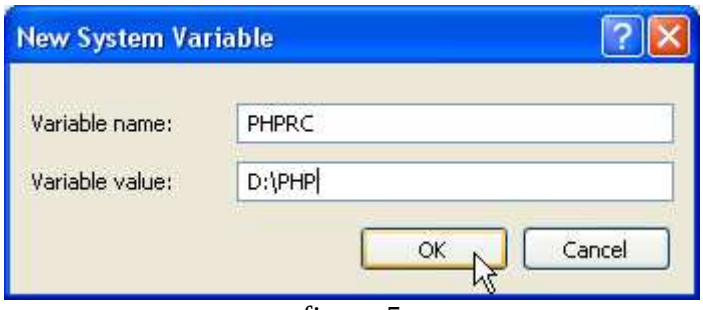

figure 5

Set the PHP Application Mapping in IIS

Before IIS will parse PHP scripts, you must first create an application mapping for PHP. This application mapping basically tells IIS what ISAPI dll, or CGI executable to use to process files of a certain file extension (in this case .php). There are two ways in which we can do this, via the command line or via the IIS MMC snap in. I'm going to show you how to do both.

As this is a universal PHP installation guide for IIS 5.0, 5.1 and 6.0 there are limited universal ways in which to add the PHP application mapping via the command line. What ever you do though **do not** use the **adsutil.vbs** script to try and add you PHP application mapping, at least not without understanding it's limitations. When you add/delete an application mapping with **adsutil.vbs** there is no way you can add/delete a single item at a time. So by using **adsutil.vbs** to add an application mapping you will effectively delete every other mapping at that level. However, what you can do is use David Wang's awesome chglist.vbs script to add the PHP application mapping without effecting existing mappings. You can download this script from David's old blog or from the bottom of this page. Once you have saved the script, simply open a command window and change into the directory that contains your script and type this in;

chglist.vbs W3SVC/ScriptMaps "" ".php,D:\PHP\php5isapi.dll,5" /INSERT /COMMIT

This will add the PHP ISAPI script map to the root of the IIS metabase, and any sites created after adding this script will automatically inherit it. If you want to add the PHP application mapping to a specific website then you will need to modify the **W3SVC/ScriptMaps** section of the command to suite. If none of this makes sense at all, don't worry because all this can be done using the GUI as well which we will cover now.

- Open the IIS MMC and right click on the **Web Sites** container (or click on any specific website you wish to add PHP to) and select the **properties** menu item

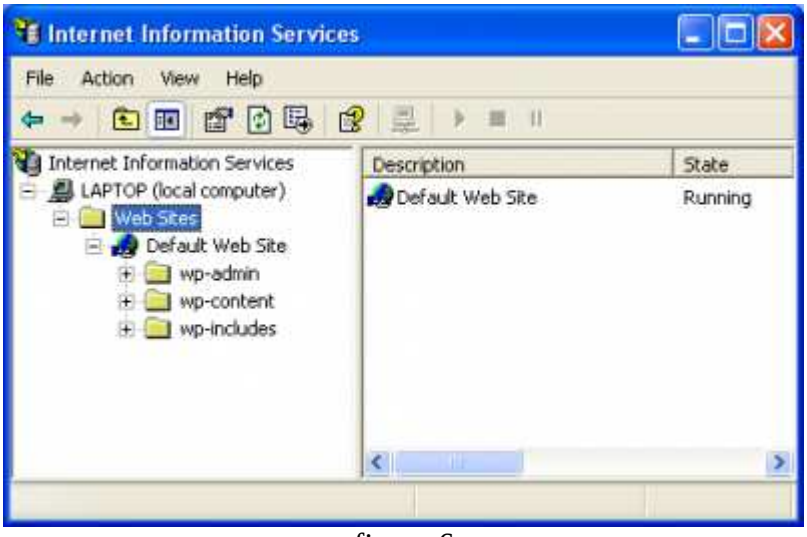

figure 6

- Once the **Web Sites Properties** window opens select the **Home Directory** tab and click the **Configuration** button as shown below in figure 7.

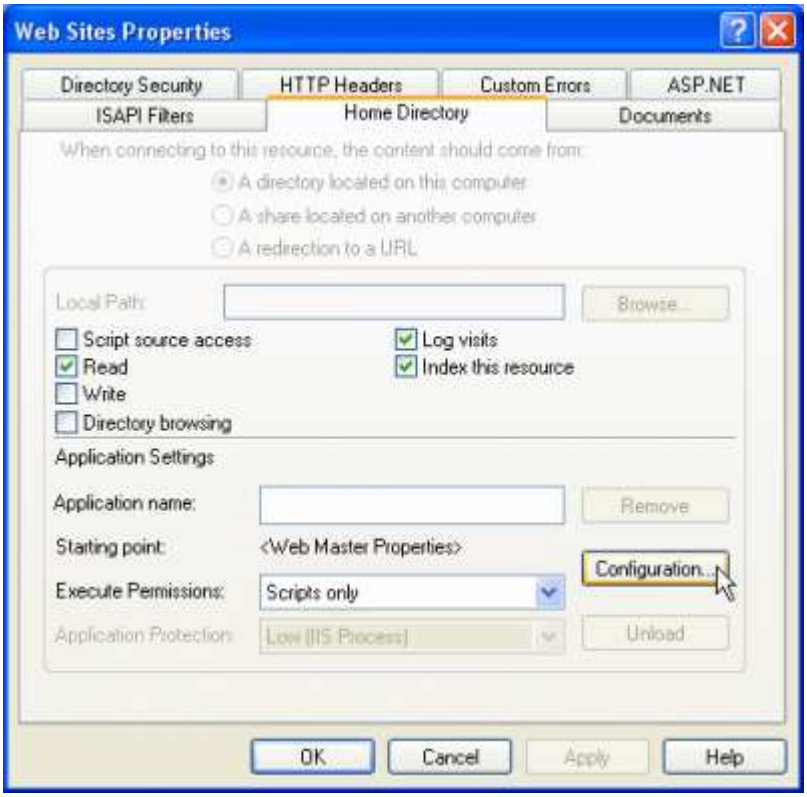

figure 7

- This will bring you to the **Application Configuration** window. Click the **Add** button at the bottom to add the PHP application mapping as shown below in figure 8.

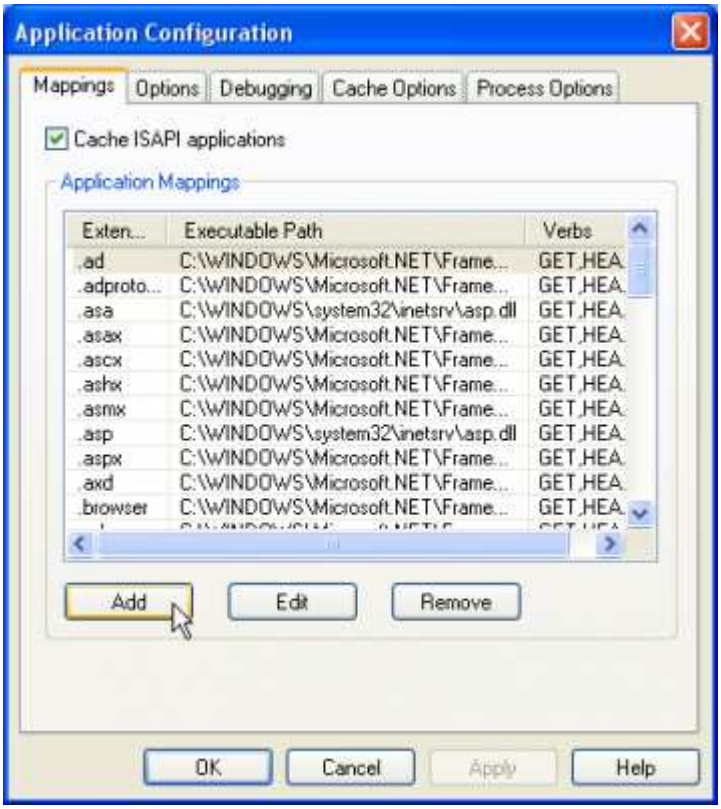

figure 8

- In the **Extension** field enter **.php** and then click on the **browse** button in the top right and select the **php5isapi.dll** file from inside you PHP installation directory (e.g D:\PHP). The default is to accept all verbs, but if you wish you can limit the verbs to tighten security for you PHP application mapping. Ensure that the **Script engine** and **Check that file exists** options as ticked and then press ok until you exit the properties windows.

figure 9

That is it, IIS should now be configured to serve PHP content. There is one exception though and that is if you are using IIS 6 then you will also need to configure a **Web Extension**, so if you are using IIS 5.0 or 5.1 you can proceed onto the next section. You can set the **Web extension** via command line using **iisext.vbs** or via the Gui. Using the **iisext.vbs** at the command prompt you'll need to type in this, but be sure to change the path to php5isapi.dll to suit your environment ;

iisext.vbs /AddFile D:\php\php5isapi.dll 1 PHP5ISAPI 1 "PHP 5 ISAPI"

To add the extension via the GUI, read on;

- Open the IIS MMC and expand the server tree. Right click on the **Web Service Extensions** option and select the **Add a new Web service extension** option.

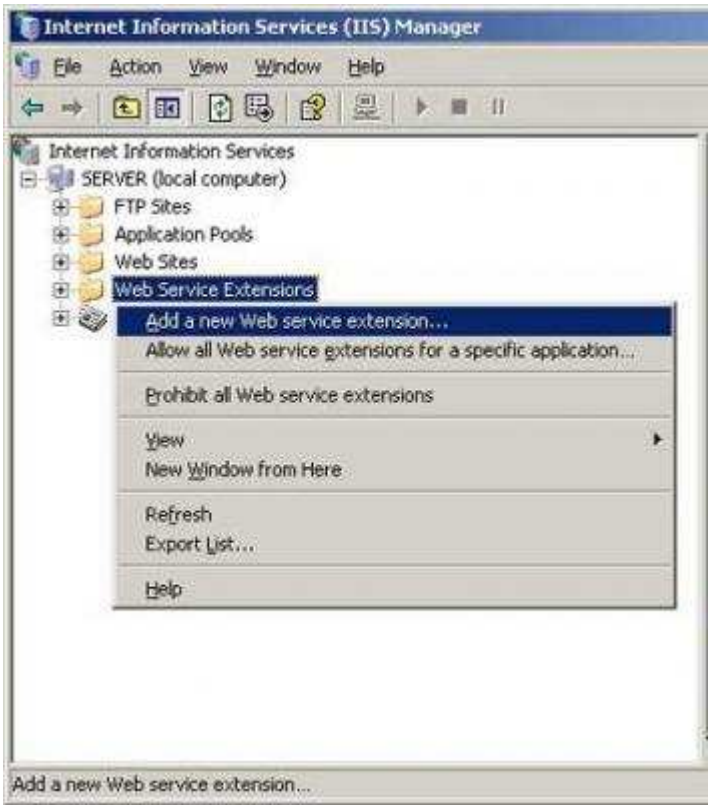

figure 10

- In the **Extension name** field put in a descriptive name, such as PHP 5 ISAPI as shown below in figure 11. Make sure the **Set extension status to Allowed** option is checked and then click the **Add** button.

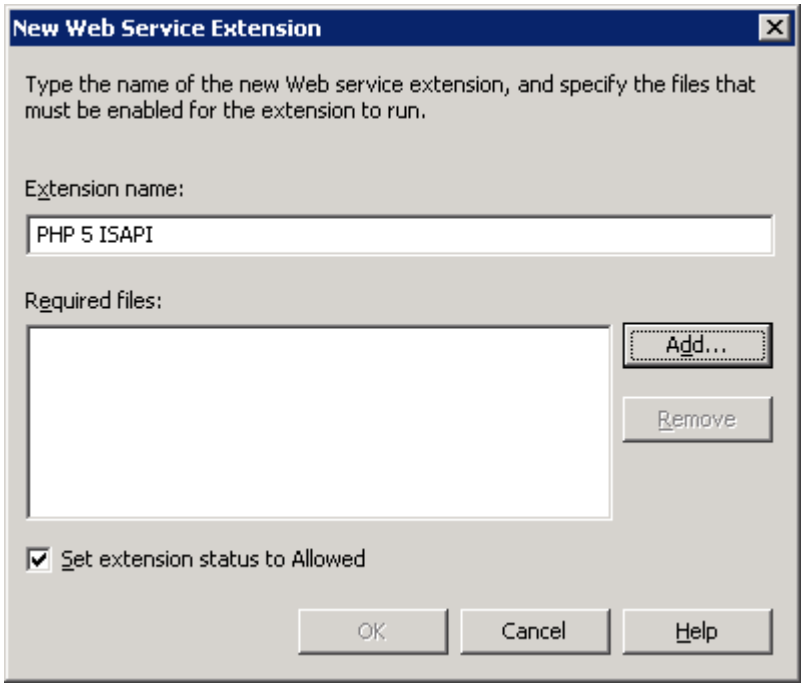

figure 11

- In the **Add file** window click the **Browse** button and navigate to your PHP installation directory and select the **php5isapi.dll** file as shown below in figure 12. Then click the **ok** button until you exit back into the IIS MMC.

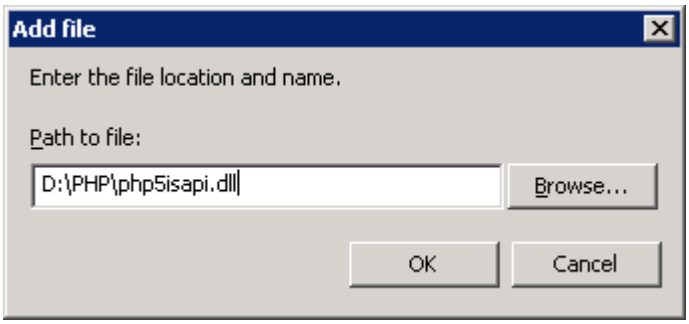

figure 12

All done, now your IIS 6 web server is all set to serve PHP content.

Configure PHP.ini

You could very easily write a series of articles regarding setting the various php.ini directives, so to try and keep this article short I will just deal with the basics here. When serving PHP on IIS the php.ini file is the only place that options are set, and if no php.ini file can be found then the defaults are loaded. The limitation of a single php configuration for an entire server is perhaps one of the biggest disadvantages with hosting PHP applications on IIS, especially in shared hosting environments.

There are actually only a handful of setting you will might want to look at configuring in php.ini, and they are the path to your extensions, sessions, and upload directories. You'll also may wish to enabled a few PHP extensions to get added functionality from PHP that might be required by some PHP apps, and perhaps set some PHP logging options.

- Open your PHP installation directory (e.g D:\PHP) and copy **php.ini-recommended** and rename the copy to **php.ini**
- Open **php.ini** and do a search for **extension_dir = "./"** and replace it with **extension_dir = "D:\PHP\ext"** where **D:\PHP\ext** is the path to your extensions directory (it's the one with all the files that begin with "php_" in it). The path to your PHP installation will be used for the next several steps, so take care that you modify them to suit your own environment.
- Search for **;session.save_path = "/tmp"** and replace it with **session.save_path = "D:\PHP\sessions"** (note the semi colon (;) is removed). This will be used to store PHP session information which is used by many PHP applications.
- Search for **;upload_tmp_dir =** and replace it with **upload_tmp_dir = "D:\PHP\upload"** (note the semi colon (;) is removed). This is used for all your HTTP upload files, and if not set your system temp will be used.
- Search for **;error_log = filename** and replace with **error_log = "D:\PHP\log\php.log"** (note the semi colon (;) is removed). This is where all your PHP logging information is kept when enabled. You can also configure

PHP to log data to your Event Log, but it requires loosening the privileges on the IUSR_COMPUTERNAME account IIS uses.

- Search for **;extension=php_mysql.dll** and remove the semi colan (;) from infront of it so it appears as **extension=php_mysql.dll**. You'll almost certainly need this PHP extension loaded as any PHP application that uses the MySQL database will need it.